The evolution of operating Systems (Types of OS)

- Must adapt to hardware upgrades and new types of hardware.
  - Character vs. graphic terminals, mouse –DVD-flash-etc.
- Must offer new services, e.g., internet support.
- The need to change the OS on regular basis place requirements on it's design:
  - Modular construction.
  - Clearly defined interface
  - Programs documentation

٨٩

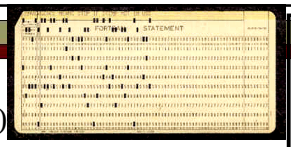
The evolution of operating Systems

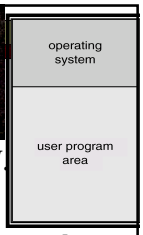
- Serial processing (uniprogramming) -Early Systems (1950)
- Simple Batch Systems (1960)
- Multiprogrammed Batch Systems (1970)
- Time-Sharing and Real-Time Systems (1970)
- Personal/Desktop Computers (1980)
- Multiprocessor Systems (1980)
- Networked/Distributed Systems (1980)
- Web-based Systems (1990)

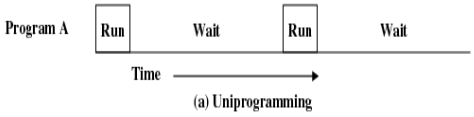
٩٠

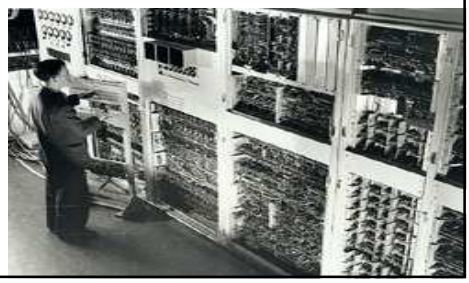
- Serial processing (uniprogramming)
  - From the late 1940-to the mid 1950)
  - The programmer interacted directly with the computer H/W.
  - There was no operating system
  - Processor must wait for I/O instruction to complete before preceding
  - The machines were run from a console consisting of display lights, toggle, switches,..
  - Paper tape or punched cards
- Early Software:
  - Assemblers, Loaders, Linkers
  - Libraries of common subroutines
  - Compilers, Device drivers
- The main problems

Scheduling management , Long setup time
Extremely slow I/O devices,
Very low CPU utilization

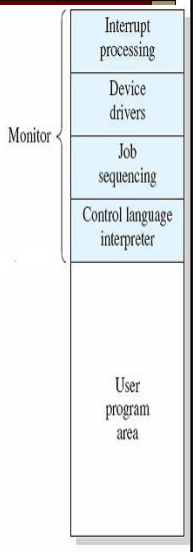






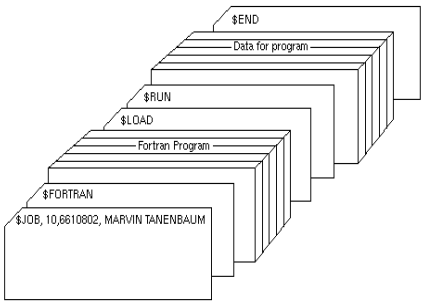


- Simple batch system
  - In the mid 1950, by General Motors for use on IBM 701)
  - The first operating system S/W was called monitor
  - The user submits the job on cards or tape to a computer operator
  - The operator batches similar jobs together and place it for use by the monitor (Reduce setup time)
  - Each program is constructed to branch back to the monitor when it completes processing
  - Monitor automatically begin loading next program
- Parts of resident monitor:
  - Control Language Interpreter – responsible for reading and carrying out instructions on the cards.
  - Loader : loads systems programs and applications programs into memory.
  - Device drivers: know special characteristics and properties for each of the system's I/O devices.



Memory Layout for a Resident Monitor

- Problems:
  1. How does the monitor know about the nature of the job (e.g., Fortran versus Assembly) or which program to execute?
  2. How does the monitor distinguish:
     (a) job from job?
     (b) data from program?
- Solution: Introduce Job Control Language (JCL) and control cards.
  - Special cards that tell the monitor which programs to run:
    $JOB - $FTN - $RUN - $DATA - $END
  - Special characters distinguish control cards from data or program cards:



٩٣

Desirable Hardware Features

- Memory protection for monitor : while the user program is executing, it must not alter the memory area containing the monitor
- Timer: prevents a job from monopolizing the system
- Privileged instructions: can be executed only by the resident monitor.
  - A trap occurs if a program tries these instructions.
    - A trap/exception is a software-generated interrupt caused by an error of the user program, for example: arithmetic overflow/underflow, division by zero , execute illegal instruction, reference outside user's memory space.
- Interrupts: gives OS more flexibility in controlling user programs

Modes of Operation

User Mode:

user program executes in user mode
certain areas of memory are protected from user access
certain instructions may not be executed

Kernel Mode

monitor executes in kernel mode
privileged instructions may be executed
protected areas of memory may be accessed

٩٤

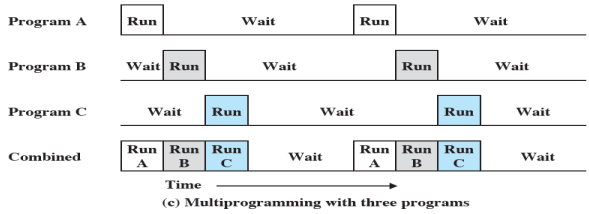
Problems

- Card Reader slow, Printer slow (compared to Tape).
  - Solution: **Offline operation**. speed up computation by loading jobs into memory from tapes and card reading and line printing done offline.
- I/O and CPU could not overlap: the CPU is often idle, because the speeds of the I/O devices are slower than of electronic devices (CPU).
- Solution: **Spooling** (Simultaneous Peripheral Operation On Line.).
  - Spool is a buffer that holds output for a device.
  - While executing one job, the OS:
    - Reads next job from card reader into a storage area on the disk (Job pool).
    - Outputs printout of previous job from disk to printer.
  - Job pool – data structure that allows the OS to select which job to run next in order to increase CPU utilization.

٩٥

- Multi-programmed and batch systems 1965-1988
  - (multi-programming or multi-tasking)
  - A mode of operations that provides for the interleaved execution of two or more computer programs by a single processor.
    - memory is expanded to hold three, four, or more programs
  - When one job needs to wait for I/O, the processor can switch to the other job
  - Allow more than one user program stored in main memory simultaneously
  - It need enough memory, memory management, job scheduling management

Problems:
Batch multiprogramming does not support interaction with users.



(c) Multiprogramming with three programs

Time-sharing systems (1980- up to now)

- The computer provides computing services to several or many user concurrently on-line (interactive).
- The various users are sharing the processor time (time slice, time slot, quantum) the memory and other resources of the computer system.
- Multiple users simultaneously access the system through terminals, with the OS interleaving the execution of each user program in a short burst or quantum of computation

Compatible Time-Sharing System (CTSS)

First time-sharing system developed at MIT (Massachusetts Institute of Technology in Cambridge)

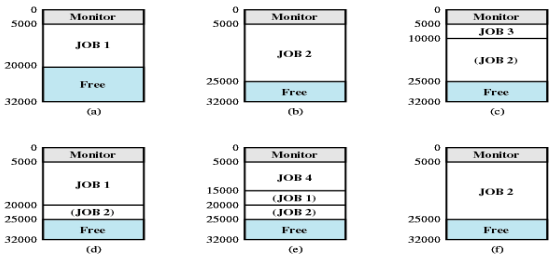


Figure 2.7 CTSS Operation

Batch Multiprogramming vs. Time Sharing

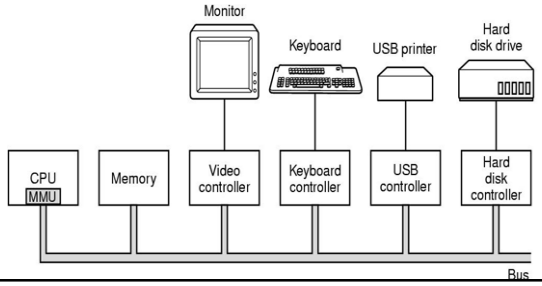
	Batch Multiprogramming	Time Sharing
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal

Real-Time Systems

- Real-Time (RT) systems are dedicated systems that need to adhere to deadlines , i.e., time constraints.
- Correctness of the computation depends not only on the logical result but also on the time at which the results are produced.
- Hard real-time system must meet its deadline.
- Conflicts with time-sharing systems, not supported by general-purpose OSs. Often used as a control device in a dedicated application: Industrial control - Robotics
- Secondary storage limited or absent, data/program is stored in short term memory, or Read-Only Memory (ROM).
- Soft real-time system: Deadlines desirable but not mandatory.
  - Limited utility in industrial control or robotics.
  - Useful in modern applications (multimedia, video conference, virtual reality) requiring advanced operating-system features.

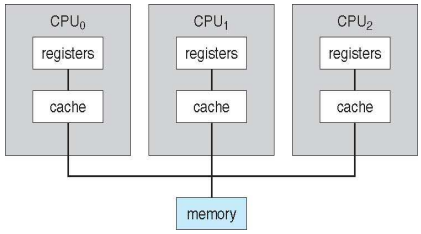
Personal/Desktop Computers

- Personal computers: computer system dedicated to a single user.
- I/O devices: keyboards, mice, display screens, small printers.
- User convenience and responsiveness.
- Can adopt technology developed for larger operating system; often individuals have sole use of computer and do not need advanced CPU utilization of protection features.
- May run several different types of operating systems (Windows, MacOS, UNIX, Linux)



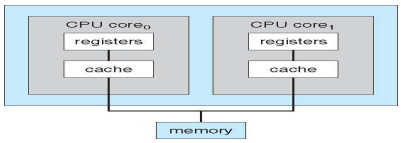
Two categories of Computer Systems

- Single Instruction Single Data (SISD)
  - single processor executes a single instruction sequence to operate on data stored in a single memory.
  - This is a Uniprocessor.
- Multiple Instruction Multiple Data (MIMD)
  - a set of processors simultaneously execute different instruction sequences on different data sets.
  - This is a Multiprocessor.



Multiprocessor Systems (OS)

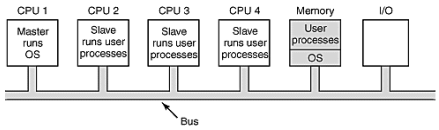
•System with several CPUs in close communication:
•processors share memory and a clock.
•communication usually takes place through the shared memory.
•Also known as parallel systems, tightly-coupled systems.

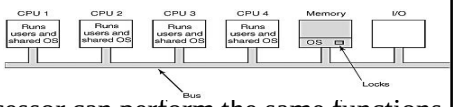


- Multiprocessors systems growing in use and importance – advantages include: Increased throughput, Economy of scale, Increased reliability , graceful degradation or fault

Types of Multiprocessor Systems Types of Multiprocessor Systems

- *Asymmetric Multiprocessing*
  - Master processor schedules and
  - allocates work to slave processors
  - the master is a bottleneck.

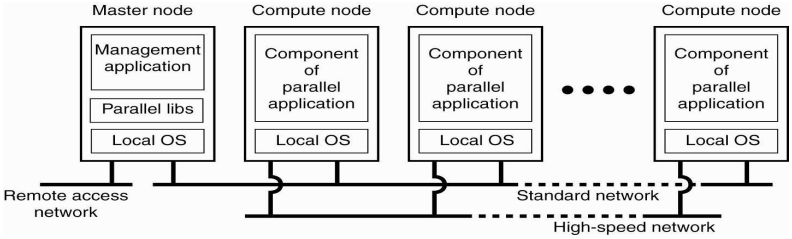




- *Symmetric Multiprocessing (SMP)*
  - All processors are peers Each processor can perform the same functions and share same main memory and I/O facilities (symmetric).
  - Are controlled by a single OS instance that treats all processors equally.
  - The OS schedule processes/threads across all the processors (real parallelism). Existence of multiple processors is transparent to the user.
  - Robustness: a single CPU failure does not halt the system, only the performance is reduced. Most modern OSs support SMP.

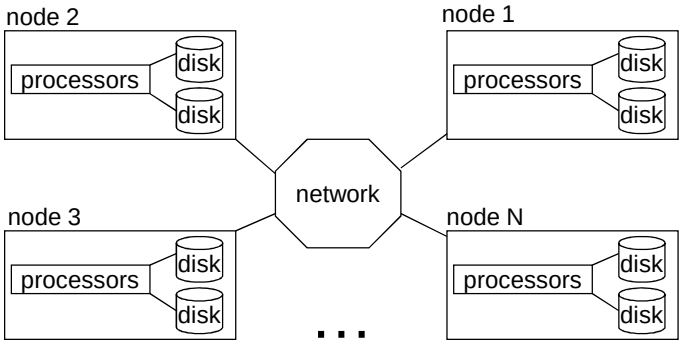
Clustered Systems (Multi computers)

- Loosely-coupled: Each processor has its own memory and copy Of the OS.
- A computer cluster is a group of loosely coupled computers connected through fast local area networks, that work together closely so that in many respects it can be viewed as though it were a single computer.
- Node is a single or multiprocessor system with a memory, I/O facilities and O/S.
- Because the processors on one node cannot directly access the memory on the other nodes, programs or software run on clusters usually employ a procedure called "message passing" to get data and execution code from one node to another.



Networked Systems

- Distribute resources and the computation among several physical processors.(*Loosely coupled system*)
- Advantages:
  - Resources Sharing -Computation speed up – load sharing -Reliability
- Most are Local area networks (LAN) or Wide area networks (WAN).
- May be either Centralized Sever or Client/Server or Peer-to-Peer (P2P) systems.



Computer Startup

- ***Bootstrap program*** is loaded at power-up or reboot
  - Bootstrap program is part of BIOS (Basic Input / Output System)
    - Typically stored in ROM(read-only memory) or EEPROM (electrically erasable programmable read-only memory), generally known as firmware

- BIOS bootstrap performs these 4 operations
  - Power-on Self-Test (POST)
    - Tests to establish which devices are present
  - Initialization of H/W devices
    - So devices can operate without conflicts on ITQ lines and I/O ports
  - Searches for OS to boot
    - Search order defined in BIOS. E.g., floppy, then any hard disk, then any CD-ROM
  - Loads the Boot Loader
    - Which loads operating system kernel and starts execution

١٠٥

Embedded OS

- Embedded OS run on the computer that control devices that are not generally thought as computers such as TV sets, mobile phones, etc.
- It is a special type of real-time systems.
- They have limitation in size, memory and power
- Examples: Android , Windows Mobile, Symbian

١٠٦

Memory Management

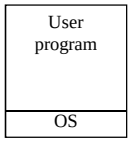
- Allocating and sharing main memory among a number of active processes.

Memory management activities

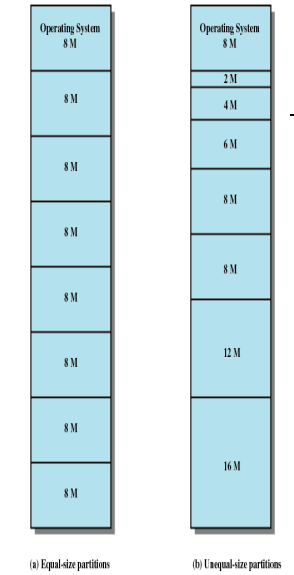
- Keep track of which parts of memory are currently being used
- Allocate and deallocate memory space as needed
- Decide which processes to be loaded into memory
- Swapping active processes in and out of main memory (relocation)
- Protect each process from unwanted interference by other processes

١٠٧

Memory Management Techniques



- Uniprogramming
  - The main memory is divided into two parts, one part for OS (monitor) and one part for the program currently being executed
  - Disadvantage: one program at a time.
- Fixed Partitioning
  - main memory is divided into a number of static partitions at system generation time
  - Equal size partition:
    - any process whose size is less or equal to the partition size can be loaded into any available partition.
    - Disadvantage: a program may be to big to fit into a partition
    - Main memory use in inefficient
  - Unequal-size partition
    - Assign each process to the smallest partition within which it will fit
    - One process queue per partition
    - Advantage:
    - Minimize wasted memory within partition (internal fragmentation)
    - Disadvantage: large partition may be unused, even when some smaller processes could have been assigned to it.



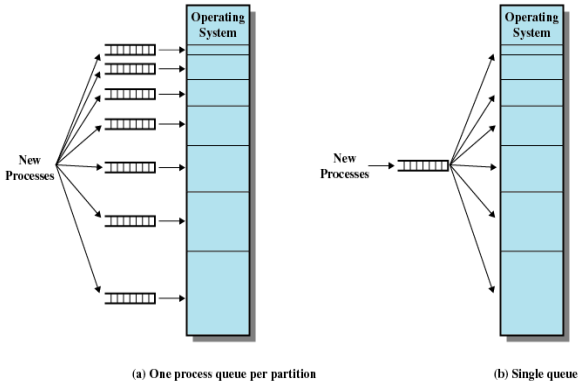


Figure 7.3 Memory Assignment for Fixed Partitioning

Figure 7.2 Example of Fixed Partitioning of a 64-Mbyte Memory

- Single queue for all processes
  - At load time, the smallest available partition that will hold the process is selected
  - Disadvantages of fixed partitioning
    - Limit number of active processes
    - Internal fragmentation
- Dynamic partitioning (variable size)
  - The partitions used are of variable length and number. When a process is brought into main memory, it is allocated exactly as much memory as it requires and no more.
  - Advantage: use memory space efficiently
  - Disadvantage: external fragmentation, the memory that is external to all partitions are a lot of small holes in memory.
  - External fragmentation solution
    - Compaction: from time to time, OS shifts the processes so that they are contiguous and so that all the free memory is together in one block

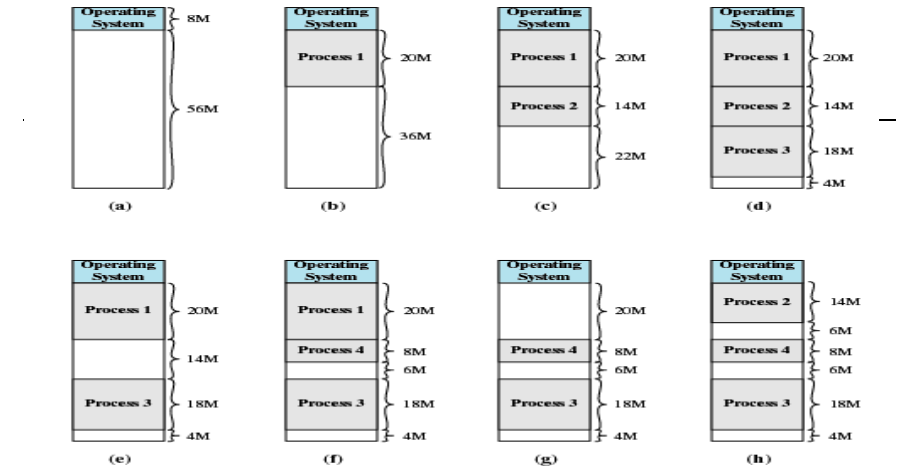


Figure 7.4 The Effect of Dynamic Partitioning

Dynamic Partitioning Placement Algorithm

- Operating system must decide which free block to allocate to a process
- Best-fit algorithm
  - Chooses the block that is closest in size to the request
  - Worst performer overall
  - Since smallest block is found for process, the smallest amount of fragmentation is left
  - Memory compaction must be done more often
- First-fit algorithm
  - Scans memory form the beginning and chooses the first available block that is large enough
  - Fastest
- Next-fit
  - Scans memory from the location of the last placement
  - More often allocate a block of memory at the end of memory where the largest block is found
  - The largest block of memory is broken up into smaller blocks
  - Compaction is required to obtain a large block at the end of memory

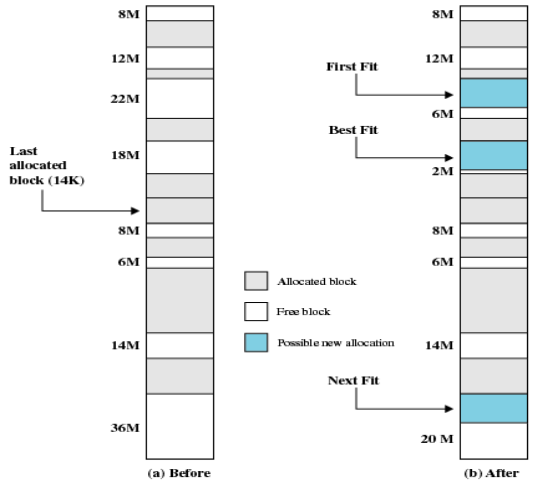


Figure 7.5 Example Memory Configuration Before and After Allocation of 16 Mbyte Block

Segmentation

- All segments of all programs do not have to be of the same length
- There is a maximum segment length
- Addressing consist of two parts - a segment number and an offset
- Since segments are not equal, segmentation is similar to dynamic partitioning

١١٤

Paging

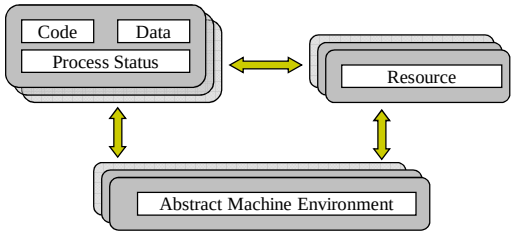
- Partition memory into small equal fixed-size chunks and divide each process into the same size chunks
- The chunks of a process are called pages and chunks of memory are called frames
- Operating system maintains a page table for each process
  - Contains the frame location for each page in the process
  - Memory address consist of a page number and offset within the page
- Combined paging and segmentation

address	Segment no	Page no	Offset no

١١٥

Threading

- A **process** is a sequential program in execution.
- A **process** is a unit of computation.
- **Process components:**
  - The program (code) to be executed.
  - The data on which the program will execute.
  - Resources required by the program.
  - The status of the process execution.
- A process runs in an abstract machine environment (could be OS) that manages the sharing and isolation of resources among the community of processes.



Program and Process

- A program is a static entity made up of program statements. The process define the run-time behavior (the execution of the program)
- A process is a dynamic entity that executes a program on a particular set of data.
- Two or more processes could execute the same program, each using their own data and resources.
- Task manager

- **A thread:** is an independent execution path, able to run simultaneously with other threads.
  - **Multithreading** : allows an application to have multiple threads of execution running concurrently. (ex, word , spell checking)
- In .NET a program starts in a single thread created automatically by the CLR (Common Language Runtime) and operating system (the "main" thread), and is made multithreaded by creating additional threads
- CLR assigns each thread its own memory stack so that local variables are kept separate
- Threads share data if they have a common reference to the same data
- All threads within a single application are logically contained within a process
- The key difference between threads and processes:
  - Processes are fully isolated from each other;
  - Threads share memory with other threads running in the same application